An Open Source Framework Approach to Support Condition Monitoring and Maintenance

Campos, Jaime; Sharma, Pankaj; Albano, Michele; Ferreira, Luis Lino; Larrañaga, Martin

Published in:
Applied Sciences

DOI:
10.3390/app10186360

Published: 01/01/2020

Document Version
Publisher's final version

applied sciences

MDPI

Article

An Open Source Framework Approach to Support Condition Monitoring and Maintenance

Jaime Campos [1,*], Pankaj Sharma [2], Michele Albano [3], Luis Lino Ferreira [4] and Martin Larrañaga [5]

1 Department of Informatics, Faculty of Technology, Linnaeus University, SE-35195 Växjö, Sweden
2 Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117602, Singapore; pankajsharma@nus.edu.sg
3 Department of Computer Science, DEIS, Aalborg University, 9220 Aalborg, Denmark; mialb@cs.aau.dk
4 School of Engineering of the Polytechnic Institute of Porto, CISTER, ISEP, 4200 Porto, Portugal; llf@isep.ipp.pt
5 BA2E06 Operation and maintenance, VTT Technical Research Centre of Finland, 02150 Espoo, Finland; martin.larranaga@vtt.fi
* Correspondence: jaime.campos@lnu.se

Received: 4 August 2020; Accepted: 9 September 2020; Published: 12 September 2020

Abstract: This paper discusses the integration of emergent ICTs, such as the Internet of Things (IoT), the Arrowhead Framework, and the best practices from the area of condition monitoring and maintenance. These technologies are applied, for instance, for roller element bearing fault diagnostics and analysis by simulating faults. The authors first undertook the leading industry standards for condition-based maintenance (CBM), i.e., open system architecture–condition-based maintenance (OSA–CBM) and Machinery Information Management Open System Alliance (MIMOSA), which has been working towards standardizing the integration and interchangeability between systems. In addition, this paper highlights the predictive health monitoring methods that are needed for an effective CBM approach. The monitoring of industrial machines is discussed as well as the necessary details are provided regarding a demonstrator built on a metal sheet bending machine of the Greenbender family. Lastly, the authors discuss the benefits of the integration of the developed prototypes into a service-oriented platform, namely the Arrowhead Framework, which can be instrumental for the remotization of maintenance activities, such as the analysis of various equipment that are geographically distributed, to push forward the grand vision of the servitization of predictive health monitoring methods for large-scale interoperability.

Keywords: OSA–CBM; rolling element bearing fault; the internet of things; arrowhead framework

1. Introduction

The need for the development of new technologies in the modern world emanates from the emerging requirements foo society and the market. Information and communication technologies (ICTs) innovations are required to support the development of new products and to deliver novel services in today's markets [1]. Each of the stakeholders in society as well as in the market place differing demands on this development. Companies need to continuously adopt novel technologies and adapt to the latest ICT developments, otherwise they are prone to suffer in terms of competitive advantage over their competitors and lose market share and profits [2]. Consequently, it becomes pertinent that the companies keep themselves abreast with the newly emerging ICTs and other technologies. It turns out that it is necessary to understand open source tools, big data analytics and mining, and other related new technologies in the specific domains of interest. The necessity of standardization for the integration of information and the components for interoperability becomes extremely important.

The software industry has been at the forefront of the introduction of standards in new developments. These standards help in the easier upgrading of system parts, increasing the number of suppliers with alternative technologies, faster technology development, and reduced prices [3]. Asset maintenance and its related fields have also been subject to such standardization attempts in the recent past. Open system architecture–condition-based maintenance (OSA–CBM) is such a standardization initiative that has resulted in the improved flexibility and interoperability of asset maintenance practices, especially for the field of CBM. CBM helps in carrying out asset-based maintenance on its condition through the monitoring of some of its parameters.

The operating costs related to maintenance can be quite high, especially in some industries where they can be the highest or the second-highest of the operating costs [4], and both over- and under-maintenance could be costly for the industry. Maintenance costs money, and over-maintenance will incur costs. Sometimes, unnecessary maintenance can also result in new faults in the machines. Similarly, maintenance can cause failures that result in significant losses due to the stoppage of production and corrective actions. These primary failures can sometimes cause secondary failures as well. Roller element bearings (REBs) are part of most of the mechanical machines being used in the production and manufacturing industries. In rotatory machines, the mechanical parts are the most exposed ones [5]. Failures of REBs have a substantial impact on the whole machinery; thus, it is of the foremost importance to do effective condition monitoring of REBs not to incur in lost production time and economic losses. It becomes a matter of utmost importance for the industry to decide on the optimal intervention timing for maintenance actions. CBM has been the major focus of attention because it has the capability to determine this intervention timing accurately. However, the development of such ICT systems is costly and time-consuming, and hence there is a need to have proficient strategies for developing such systems. Open source solutions are a very potent tool in such scenarios due to their lower costs, and their usability is a critical characteristic for their adoption.

When it comes to automated manufacturing or process plants, CBM is the preferred approach when it is feasible to implement it, i.e., considering both technical and financial aspects. The crucial part of the CBM is condition monitoring. It involves data acquisition, and the processing, analysis, and extraction of useful information. It also includes such aspects as the diagnostic and prognostics of the equipment. All the former mentioned to understand the current condition and predict its remaining lifetime of the machinery.

However, it is well known that complications are faced when implementing the full circle of CBM [6]. These issues are connected with, for instance, the amount of data that is produced, data integration features, as well as a lack of experts. There have been many attempts to resolve these issues with the help of different kinds of ICTs over time, such as artificial intelligence, distributed intelligence, etc. [6–8]. Currently, with the emerging technologies such as the IoT, big data and analytics, open source solutions as well as open platforms, the entrance barrier to advanced maintenance solutions has been lowered considerably. From a developer perspective, there are a number of frameworks as well as programming languages that can provide benefits, since they were used by the community to develop both general purpose and specialized solutions to tackle many use cases that comprise machine learning and CBM. One example of these languages is Python, whose mature codebase brings new aspects into the developments of these maintenance tools, like, for instance, lower development costs. All this brings new possibilities into the domain to solve the issues with, for example, data volume, its integration, and the lack of experts in the field.

Thus, such emerging ICTs as the Internet of Things (IoT) are revolutionizing several areas, including maintenance [8]. The rolling element bearing involves vital indicators to determine the machinery condition as well as its remaining lifetime in contemporary production machinery [9]. Therefore, the use of emergent technologies, such as the IoT and predictive health monitoring tools that support the rolling element bearing condition is crucial for the industry.

Our Contribution

This paper points focuses on selected perspectives that are important for these specific domains, namely for industrial maintenance engineering in particular and asset management in general. The authors, therefore, build over existing works on CBM by introducing the usage of open source solutions; paving the road to interoperability by adhering to mature standards (e.g., OSA–CBM); promoting efficiency and improving interoperability even further by the servitization of predictive health monitoring methods in connection with the promising ICTs such as the IoT. The social and technological benefits of our approach comprise the democratization and the wide acceptance of CBM.

The driving force that led to the definition of the architecture, and the creation of the prototype, was to allow for a set of benefits to be enjoyed by CBM activities. First of all, usually, the analytics activities needed for CBM need to be executed in a computing center or on the cloud, and most implementations have to create their own solution to move the data from the factory to where the data can be used; our work aims to demonstrate how to allow this by means of the Arrowhead Framework. Second, many IoT devices have limitations in terms of computational capabilities, memory, and it is cumbersome to update their software, leading to security concerns; we will show that the devices can receive seamless security by being integrated into an Arrowhead local cloud. Third, evolving and extending an industrial solution is usually challenging, since a number of subsystems need to be updated to enable the communication of new devices in the field; by means of the Registry service and the Orchestration service, which are part of the Arrowhead Framework, new devices will be easily incorporated in the solution. It has been chosen to include the rolling element bearing (REBs) fault analysis module in our Arrowhead local cloud since it is a crucial part of condition monitoring (CM) and CBM, as mentioned above. However, in the area of CM and CBM, other related modules can be included in the solution, such as machine learning algorithms that have been tested in the domain [10]. Other modules that can be incorporated into the framework are, for instance, computerized maintenance management system, among other. Finally, in most cases, it is hard for a new company to design a CBM solution since its design, implementation, and deployment all require considerable effort; as discussed previously, by focusing on open source technologies, mainstream programming languages and well accepted software architectures, it is possible to lower the entrance barrier of new actors in the industry.

The paper is organized as follows: Section 2 introduces the open system architecture and condition–based maintenance (OSA–CBM) and Machinery Information Management Open System Alliance Common Relational Information Schema (MIMOSA CRIS). The Section also describes the predictive methods. Section 3 presents the Internet of Things (IoT) frameworks from the literature. The Arrowhead Framework and its benefits are presented in Section 4, and we delve deeper into how it can help for the servitization of predictive health monitoring methods. Implemented data analysis testbeds that were incorporated into the Arrowhead local cloud are presented in Section 5. Conclusions are drawn in Section 6.

2. The Open System Architecture—Condition-Based Maintenance

Both the military and the industrial domains are showing a wider acceptance of CBM. CBM systems have to cope with a complexity that would scare most system engineers since they span over very different hardware (computers, and embedded systems such as sensors) and software that must be able to operate together to power CBM applications. The main approaches that are relevant for this work are MIMOSA CRIS and OSA–CBM.

The Machinery Information Management Open System Alliance MIMOSA consortium, founded in 1994, developed the MIMOSA CRIS standard, which aims to facilitate the exchange of information between actors in the maintenance industry. To this aim, a number of open international agreements (standards) were developed, targeting different aspects of machine maintenance [3]. In particular, the most common concretization of the standard is a relational database model for different data types related to CBM applications, and it was also the main driver for the structure of the system

interfaces and database schema itself. From the point of view of licensing and reach, MIMOSA's interface definitions are open and portray a standard data exchange for modern CBM systems.

OSA–CBM is the result of an industry initiative that had the goal of developing and demonstrating an open system architecture for condition–based maintenance. Among the stakeholders that developed the approach, there was the US Navy, and the MIMOSA consortium. The resulting platform is a de facto standard that covers all functional requirements related to this domain, such as data collection and the recommendation of maintenance actions. Modern ICT technologies allowed us to apply distributed software architectures to CBM and to make it more cost-effective than traditional standalone systems. Figure 1 represents the seven layers of architecture of OSA–CBM [11].

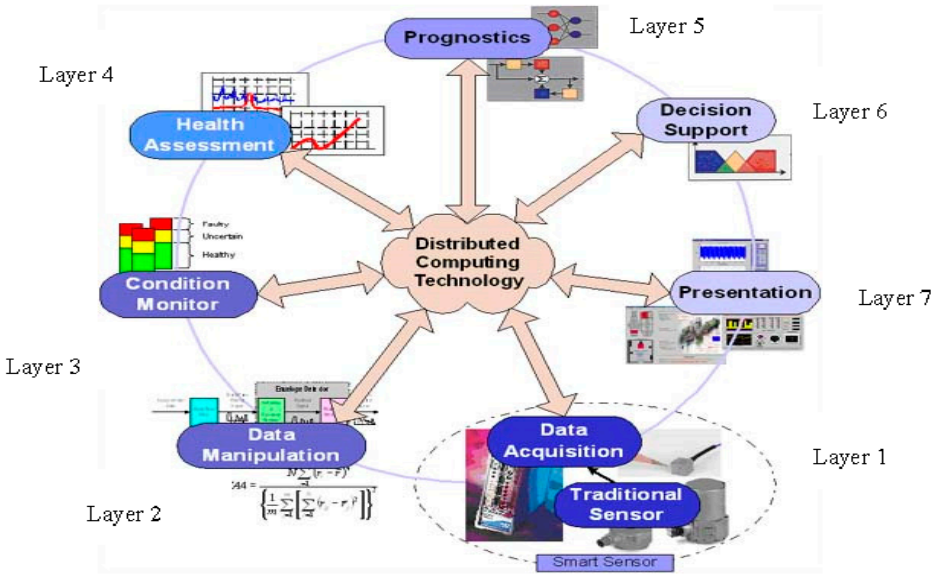

Figure 1. Data flow within an open system condition-based maintenance (CBM) design [11].

For the sake of argumentation regarding the benefits of OSA–CBM to the problem at hand, we will now describe the layer that the platform is composed of:

- Layer 1—Data acquisition: This layer is devoted to collecting data from physical sensors, digitalizing it, and passing it upstream;
- Layer 2—Data manipulation: Data are pre-processed by means of signal processing techniques while converting them in a format that is more suitable for analysis in later stages;
- Layer 3—Condition monitor: This stage looks for outliers by comparing processed data with expected data. If a threshold is reached, this layer generates an alert for the user;
- Layer 4—Health assessment: Data are received from downstream, and other health assessment modules and the layer determine the level of machinery deterioration. If this is the case, relevant fault conditions are computed;
- Layer 5—Prognostics: Collected data are processed together with statistical and historical data, to compute a projection of the machinery's remaining useful life (RUL);
- Layer 6—Decision support: This layer acts a recommender system that suggests an action among a set of alternatives, most of them usually being maintenance actions;
- Layer 7—Human interface (Presentation layer): This layer takes care of displaying to the user all the information computed by the lower layers, such as the health of the machine (Layer 4), the prognosis of its condition (Layer 5) and the recommendations for maintenance actions (Layer 6).

CBM has become the go-to methodology for the health monitoring of machines in the automated production and processing industries. However, it is a complex process that requires the collection and analysis of data from the selected asset, and the process is customized to the type of asset [12].

A CBM system collects the machine data through the embedded sensors (or other offline data collection methods) and sends them to the central server. These collected data can either be the event data or the condition monitoring data. Event data are a record of the information related to various events of the machine. These events can either be the happenings to the machine, for instance, fault occurrence, overhaul of the systems, etc. or the actions were taken by the maintainer on the machine (e.g., part replacement, oil and lubricant change, etc.). On the other hand, the condition monitoring data are those that are actually picked up by the sensors to reflect the health of the machine [13]. The most critical part of a CBM program is to decide the measurement parameters and the location where the sensors must be placed. Once the CBM system is in place, data acquisition is the first stage. The collected data from the physical asset are passed to the next stage of data processing. Data processing in itself is a two-stage process that involves the cleaning of data as well as the analysis of them. There are numerous methods and techniques that are used to enhance the quality of the data. The analysis part of the process is undertaken only after the quality of the data has been improved to acceptable levels. Figure 2 is a pictorial representation of the stages in asset health monitoring.

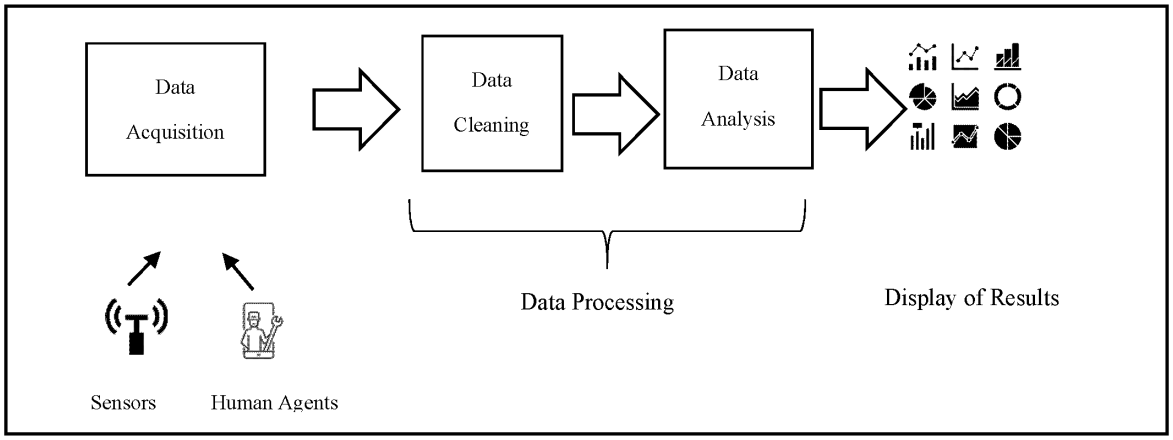

Figure 2. Data operations in the asset health monitoring process.

Diagnostic and prognostic are the two main types of health monitoring processes. Diagnostic monitoring is the process of identifying the faults after they have occurred, and the machine has broken down. It can be termed as the postmortem of the events where the fault is detected after it has already happened. Prognostics is the technique which prevents failures by predicting the likely future time of their occurrence. In medical terms, it is akin to conducting surgery to save the patient from dying. Instinctively, it can be seen that prognostics has more promise than diagnostics because it can prevent the faults and thus ensure that the machines continue to operate as intended. If the machine is allowed to run to failure, it can cause catastrophic damage by causing secondary failures in the machine. Prognostics is also useful in cutting down the logistical delays that often happen if the machine fails unexpectedly. Prognostic systems are at least able to predict a future failure, if not prevent it. This helps the maintainer in the pre-positioning of spare parts, tools, and maintenance personnel, which can eventually reduce the diagnosis and repair times. It is evident that one cannot do-away with diagnostics. Even if a good prognostic system is in place, it cannot accurately predict everything all the time. There will still be occasions when the machines fail without any warning from the prognostic system. At such times, one has to rely on diagnosis techniques to identify the fault and conduct repair on the machine by either replacing the component or repairing it. Prognostic techniques derive knowledge from the historical data to identify how the asset deteriorates before failure. This helps in charting the route on which the asset will progress, leading to the failure, and also tells the changes in the measured parameters as the asset follows this path. Health monitoring helps in identifying where on this path currently is the physical asset. In combination, these two can predict the future failures of the asset being monitored.

The data-driven prognostic models use artificial intelligence (AI) and are given the historical data as input. These data may come from CBM, Supervisory control and data acquisition (SCADA) measurements, etc. [14,15]. These prognostic models use one of the following techniques or AI algorithms for fault prediction: particle filtering, simple trend projection model, data interpolation using artificial neural networks (ANN), regression analysis and fuzzy logic, time series prediction model, exponential projection using ANN, recursive Bayesian technique, Hidden Markov Models, hidden semi-Markov model, etc. [16,17]. The emergence of big data and their various methods of analysis has introduced newer methods like clustering and classification for the prediction of faults.

3. IoT Frameworks

IoT technology is expected to offer new promising solutions in various domains and, consequently, impact a wide variety of aspects in our daily lives. Song et al. [17] describe how IoT can help in the development of smart cities. Jeschke et al. [18] give a brief introduction of the development of the Industrial Internet of Things (IIoT) also introducing the Digital Factory and cyber-physical systems. This work also presents the challenges and requirements of IIoT with the potential regarding the application in Industry 4.0. More about the Cyber-Physical systems can be read in Song [19]. IoT envisions a reality of pervasive connectivity, in which all things will be connected to the Internet and interact with the physical environment through sensors and actuators, collecting and exchanging data to feed services and intelligence, resulting in a fusion between the physical and digital worlds [20]. A plethora of platforms was created on top of IoT, and the approach was collectively named IoT platforms. Great hype was generated regarding IoT platforms [21] since it promises to be instrumental in the modernization of many industries. As the technology is still in its infancy, the development and testing of software applications and services for IoT systems encompass several challenges that existing solutions have not yet addressed.

Gyrard et al. [22] designed and developed the Machine-to-Machine Measurement (M3) framework which assists IoT developers and end-users in semantically annotating Machine to Machine (M2M) data and generating cross-domain IoT applications by combining M2M data from heterogeneous areas. It also helps in interpreting the generated data. The framework is interfaced to the end-users through the application layer. The users can peruse the results through this layer.

Vogler et al. [23] presented LEONORE, an infrastructure and toolset for provisioning application components on edge devices in large-scale IoT deployments. The authors propose an approach where the installable application packages are fully prepared on the provisioning server and specifically catered to the device platform to be provisioned. This ensures that the resource constraints of IoT gateways are taken care of. The solution is demonstrated to have both the push- and pull-based provisioning of devices. In pull-based provisioning, the devices are allowed to independently schedule provisioning runs at off-peak times, whereas push-based provisioning allows for greater control over the deployed application landscape by immediately initiating critical software updates or security fixes. The authors illustrate the feasibility of the solution using a testbed based on a real-world IoT deployment.

The testing of IoT devices is a topic of interest for academia as well as industry. This IoT testing takes the lead from the traditional software industry and includes interoperability testing and conformance testing. The amount of IoT devices and their collaborative behavior causes new challenges to the scalability of conventional software testing, and the heterogeneity of IoT devices increases the costs and the complexity of the coordination of testing due to the number of variables. Kim et al. [24] present a new concept called IoT Testing as a Service—IoT-TaaS, a novel service-based approach for an automated IoT testing framework. The paper first conducts an analysis of traditional software testing methodologies, i.e., conformance and interoperability testing, in order to retrieve key challenges of IoT testing. Thereafter, an insight on how traditional testing methodologies have to be evolved to provide a testing framework adequate for IoT testing is given. The proposed method aims

to resolve constraints regarding the coordination, costs, and scalability issues of traditional software testing for a standards-based development of IoT devices.

Kim et al. [24] propose to employ Topology and Orchestration Specification for Cloud Applications (TOSCA) for IoT service management. TOSCA is a new standard aiming at describing the topology of cloud applications by using a common set of vocabulary and syntax. TOSCA helps in improving the portability of cloud applications in the face of growingly heterogeneous cloud application environments.

Pontes et al. [25] describe a pattern-based test automation framework for the integration testing of IoT ecosystems and call it Izinto. The framework implements a set of test patterns specific to the IoT domain, which can be easily instantiated for concrete IoT scenarios with minimal technical knowledge. This paper also demonstrates its validation in a number of test cases within a concrete application scenario in the domain of ambient assisted living (AAL).

Jararweh et al. [26] proposed a comprehensive Software-Defined IoT (SDIoT) system architecture solution to accelerate and facilitate IoT control and management operations, and at the same time, address the issues of the classical design. The paper highlights how the software-defined system handles the challenges of traditional system architecture as it provides a centralized, programmable, flexible, simple, and scalable solution to control the systems. The paper presents an SDIoT architectural model along with its main elements and shows how these elements interact with each other to provide a comprehensive framework to control the IoT network.

In conclusion, there are several efforts to develop IoT automation solutions and frameworks for various areas. In the field of condition monitoring and maintenance, which is the result of a set of European projects, namely MANTIS, Arrowhead, SOCRADES, IMC-AESOP, ARUM, INTER-IoT, etc., in which service-oriented architectures (SOA) principles have been applied to IoT and industrial applications. More through comparison between Arrowhead and competing approaches is provided in [27] and in [28]. In the next section, details about the Arrowhead Framework are presented.

4. The Arrowhead Framework

The Arrowhead project (2014–2017) was a European effort aimed at applying SOA to IoT platforms supporting industrial automation scenarios. The use cases of the project spanned from manufacturing to energy optimization to machinery maintenance. A number of other projects (Productive 4.0, MANTIS, Arrowhead Tools, SOCRADES, IMC-AESOP, ARUM, INTER-IoT, etc.) pushed further the results of the Arrowhead project, to extend the coverage to other use cases and to make the technology more user friendly to use and more adaptable to new technological challenges.

The main tenant of the Arrowhead project is the application of SOA [1]. In fact, the Arrowhead approach considers that all interactions are mediated by services. Services are produced and consumed by systems, which are executed on devices. Services and systems can be application ones when implementing a business case, or core ones when they provide support to other systems and services. Examples of application services can allow for sensor readings, controlling AC devices, obtaining energy consumption, etc. The core systems are usually shipped together as the Arrowhead Framework. Systems register themselves and the services they produced on the Service Registry core system and use the Orchestrator system to look up the systems that can produce the services they need. The Orchestrator, as the name implies, is able to provide a set of services that are then used together to produce a composed- and more complex-service. Moreover, the Arrowhead Framework takes care of security by means of the Authorization core system, which authenticates the systems and provides them with a token to access the services they need.

The deployment unit is called a local cloud, which is an autonomous set of devices that have access to an Arrowhead Framework. A minimal local cloud is represented in Figure 3, where the service producers and consumers are represented with the usual schema: the endpoint of a line that represents an interaction ends with an O for a service producer, and with a C for service consumption.

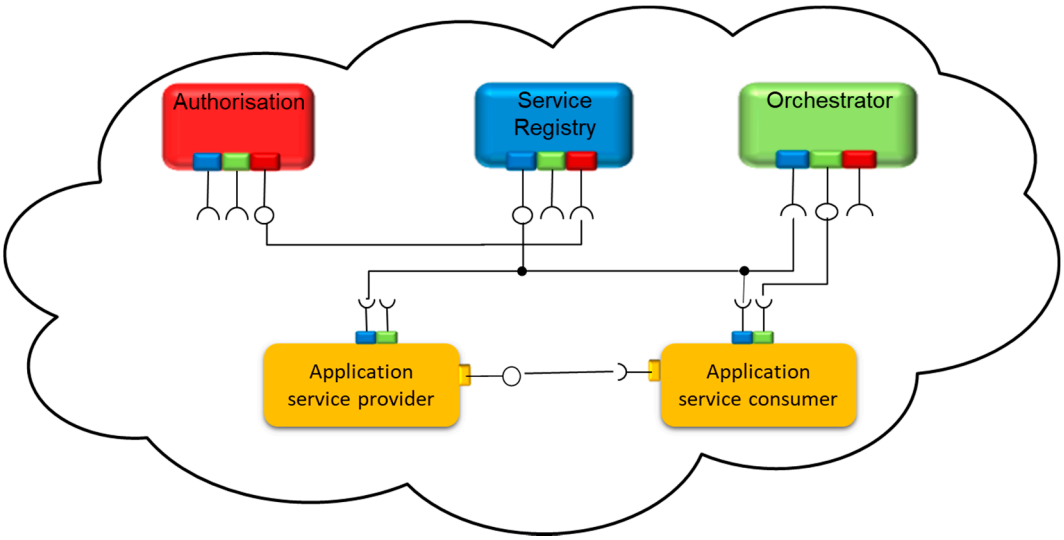

Figure 3. Minimal arrowhead local cloud.

Multiple local clouds can exist and be used to create a distributed application. Each local cloud will have its own Arrowhead Framework, and it will use a Gateway core service to create a secure tunnel between the local clouds, which will permit a system to consume a service provided in another local cloud. The management of security between local clouds is bestowed upon the Gatekeeper core service, which maintains the security token used by Gateway core systems to interact. Figure 4 represents a set of local clouds that interact through gateways.

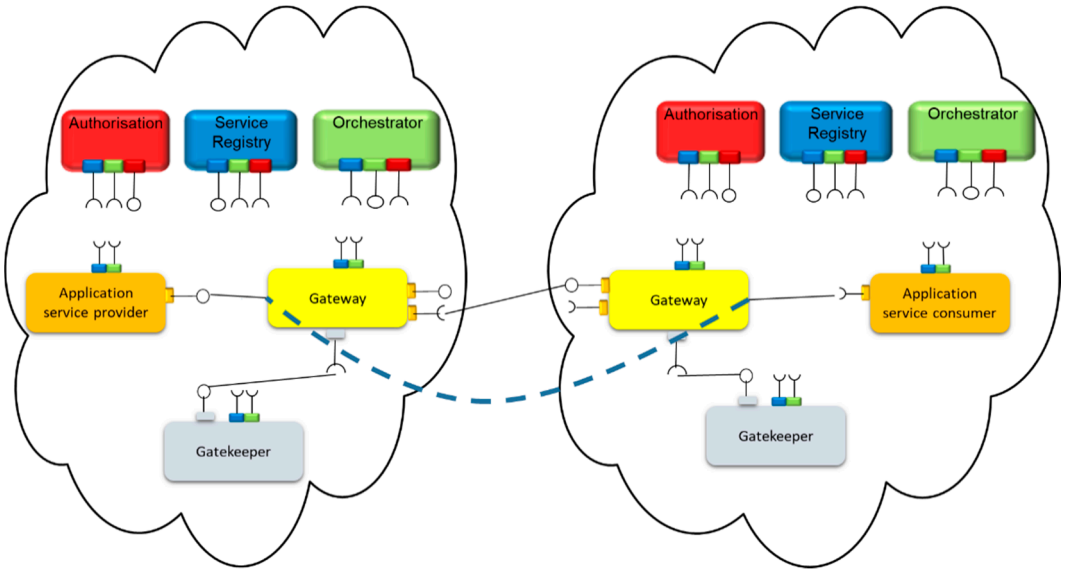

Figure 4. Inter-cloud service consumption supported by Arrowhead Frameworks.

An important benefit provided by the Arrowhead projects is a comprehensive documentation system [29] that supports the design and the maintenance of distributed applications. This is one of the key features that allowed the Arrowhead approach to become a strong player for the interoperability of industrial systems. In fact, the documentation system allows to communicate unambiguously the structure of services and the capabilities of systems, thus allowing for the creation of systems of systems.

Even though the Arrowhead approach adheres to the SOA tenants, it is not limited to REST, and in fact, existing pilots are using alternative protocols and approaches such as the Constrained Application Protocol (CoAP), OLE for Process Control—Unified Architecture (OPC-UA) and Extensible Messaging and Presence Protocol (XMPP). The number of programming languages used to create Arrowhead-compliant systems is ever-growing, and already comprises Java, Python, and C, and the Arrowhead community provides and maintains libraries [30] and software generators [31] to speed up the development process. Moreover, Arrowhead can be considered a strong player in the interoperability arena. Apart for the aforementioned protocols, the Arrowhead Framework comprises adapters to interoperate with a number of other frameworks such as Basissystem Industrie 4.0 (BaSys) [32], which is an SOA middleware for the dynamic management and reconfiguration for industrial-scale production facilities, FIWARE [33], widely adopted in Europe to build smart solutions, and IEC61499 [34], which is an international standard to define function blocks for industrial process measurement and control systems. Thus, by choosing to integrate a new system into an Arrowhead local cloud, it is possible to consume and produce services from other platforms.

The Arrowhead approach can provide solid benefits to maintenance applications since it can provide seamless security, good support for dynamic use cases, and for remotization, in which contexts where the involved systems can evolve over short time frames, and cannot always be designed prior to creation.

With regard to security, in an Arrowhead platform, a system can be authenticated and authorized to interact only with required systems and services and for the required time span. This allows for more stringent security when for example a system is from a third-party stakeholder, or when a system is executed on a mobile device that was physically part of less secure environments, or when a system pertains to a different OSA–CBM layer of the one it targets, and then it must be granted a short-term authorization to perform its functions.

Dynamicity and churning can be a problem in realistic scenarios since maintenance subsystems can be replaced, and then they must be able to register themselves, discover and be discovered, and in general, substitute previous systems. The SOA tenants that are the base of the Arrowhead approach allow developing systems independently from each other, with the reasonable confidence that they will interoperate even when being shipped by different stakeholders.

Remotization is a novelty for modern maintenance solutions, but it became a requirement for most advanced solutions. In fact, it is hardly possible to co-locate a computational cloud with factories, and thus it is important to have access to a secure inter-cloud to connect different OSA–CBM layers and deployment tiers. Arrowhead local clouds are inherently autonomous but can create secure tunnels between each other to transport data and provide capabilities between different tiers such as factory, platform, and business [12].

Another advantage of adhering to the Arrowhead approach is related to its open source nature, which guarantees a lower entrance barrier for SMEs that want to employ its technologies in their solutions. Finally, Arrowhead is part of the Eclipse project, which ensures a proper level of stability and maturity.

5. The Solution

Figure 5 illustrates the high-level conceptual model of the architecture proposed, which is based on MIMOSA. On the left part of the figure, we are representing Arrowhead Field level Local Cloud 1, which supports the communication between IoT sensors and machines with MIMOSA compliant databases. The data are related to multiples sources, such as bearing data or machine status data collected from their I/O ports. This local cloud contains the mandatory Arrowhead services, which in their turn support the producers and consumers of data by means of, for example, the Service Registry system, to allow to search for the data producers and consumers as well as the Authorization system, which contains the security credential for each one of them. Finally, the rules on how to connect application services are contained on the Orchestration System. The local cloud division allows to

effectively isolate this part of the network from the rest, which is usually composed of devices with low performance, low-security capabilities and where the update of its software can be difficult or even impossible, therefore it is of paramount importance to protect this part of the system.

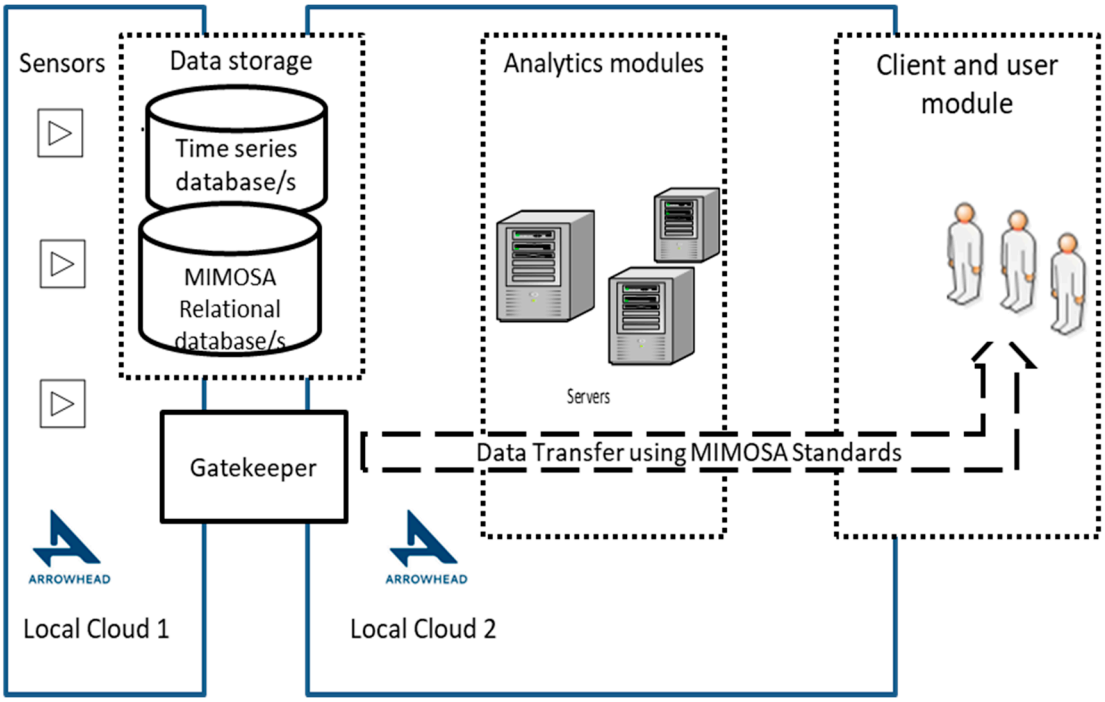

Figure 5. A conceptual model of the solution, based on MIMOSA.

These sensors store their data in the Time Series Databases, which are more capable of withstanding the data rates of the sensors. These data are then processed and stored on structured MIMOSA-compliant databases, ready for analysis.

Business logic and analytics services are hosted on server computers where different machine learning algorithms can be executed effectively, generating analytical data for higher-level decision-making. Finally, the last tier is the client machines, where the visualization of data can be performed by maintenance engineers. This part of the system is also isolated through an Arrowhead local cloud (Local Cloud 2), with the same set of functionalities as defined for Local Cloud 1. This Arrowhead local cloud supports different protocols. Although the local cloud concept isolates the two networks, it also provides through its Gatekeeper system the capability for the maintenance engineers to access online data from the sensors, e.g., for the testing of machines. In fact, the communication, mediated by the Gatekeepers, can flow between the systems in the two local clouds in a transparent manner.

The proposed approach aims to leverage open source technologies to lower the entrance barrier to advanced maintenance. Current developments on all the modules required for the solution allowed to focus on the development environment of the Python programming language. In fact, many APIs were developed by the community to serve a plethora of purposes in Python, spanning from numerical analysis to high-performance computation to secure communication and integration into an Arrowhead solution.

Our prototype was built on top of existing open source libraries to create a preliminary ecosystem of services to realize the maintenance application. For other components, our novel system was integrated with existing open source components developed in other programming languages and around other frameworks. This was facilitated by the SOA approach, which empowers components with inherent

interoperability based on the formal definition of service interfaces between the interacting systems, which is one of the tenants of SOA in general, and of the Arrowhead approach in particular.

The discussion in this section also aimed at showing that this architecture was able to provide the four benefits of our general approach: (i) the remotization of the different activities required by a CBM platform by means of inter-cloud communication, (ii) seamless security by isolating the devices into Arrowhead local clouds to protect potentially vulnerable devices from the Internet, (iii) support for dynamic use cases by means of the Service Registry/Orchestration services of the Arrowhead Framework, and (iv) democratization of CBM by means of the lower entrance barrier of open source technologies, the usage of a mainstream programming language (Python), and the high level of interoperability and easier system integration granted by SOA.

The prototypes presented next are part of the ecosystem illustrated, capable of gathering data, which are later stored and continuously accessed to be processed and conduct analytics. Lastly, the processed data are presented to the clients through human–machine interfaces providing information to support maintenance decisions.

5.1. Monitoring of Industrial Machines

The current section outlines a case study that adopts the architecture presented in Figure 5 to monitor industrial machines, gathers and transfer sensor data with the support of an efficient message-oriented middleware, which is responsible for the transport of data to the Time Series databases and to a Mimosa database/s, from where the data can be analyzed.

This case study focuses on a metal sheet bending machine of commercialized by ADIRA.

The machine is a mixed construction that is powered by both a hydraulic and electric system. The hydraulic drive contains two cylinders placed on a combination of bars that serve as actuators. These actuators move a ram vertically up and down onto a kick on the machine's base. The ram holds a punch. The workpiece is put between the punch and the pass on, acting as clamps, so that it can be distorted.

The system architecture for this system is divided into two main parts. One is the local network hosted on a machine within the factory, and the other is on the cloud. This cloud also exposes a human–machine interface (HMI), accessible via the Internet.

The components that compose the Local Cloud 1 are the press brake machine, the MANTIS-PC, the edge local, and a local, restricted capability, HMI. The press machine comprises several data sources, which extract data from its computer numerical control (CNC) and a safety programmable logic controller (PLC), and also from some other sensors which were specifically added to support the maintenance platform. The latter includes Arduino-based motes with accelerometers that monitor the ram for both acceleration and vibrations, and it is capable of detecting wear problems on the bending operations and on the mechanics of the ram system itself.

The component called Edge Local mediates the communication between Local Cloud 1 (the factory) and Local Cloud 2 (the cloud). It acts as the OPC-UA client in the interaction with the OPC-UA servers located on the MANTIS-PCs. This component also performs the pre-processing of the collected data, and it exposes the only connection towards the cloud using the Advanced Message Queuing Protocol (AMQP) protocol and the facilities provided by the Arrowhead inter-cloud communication support.

Some CBM capabilities are provided locally by cloud 1, since it contains a Local HMI, which is a simple application that the user can use to view and monitor the information available in the factory, which comprises at least the raw data collected, and the pre-processed information computed by the Edge Local.

Local Cloud 2 contains all the applications that perform high-level computation to support CBM. The cloud processes data from multiple factories, which were produced by local clouds in the factories and which were sent by edge local components. The entrance to Local Cloud 2 is through the edge server component, which receives the data from the edge local component and passes them to the number of applications executed in the cloud and that, together, compose the data analysis component.

Moreover, both the data received by the edge server and the information composed by the data analysis component are sent to the visualization component (HMI component).

The edge server component comprises a communication middleware, a MIMOSA complaint database module, and the communications server for the HMI. The primary purpose of the edge server component is to feed data to all other components in the cloud.

The middleware, which is based on an AMQP event-based messaging bus, has a central role managing the data streams from the factories, by storing and transporting the data between the edge local, the data analysis and HMI components.

The data analysis components include a set of three modules. The prediction models are used for the detection, prognosis, and diagnosis of machine failures. The models are designed for a specific machine family or generic, which can be adapted to different machine families.

The prediction application programming interface permits clients to ask for predictions from the models, taking into account the available data and the training data (which is also inserted through this interface). Internally, this part of the system is built of the Intelligent Maintenance Decision Support System (IMDSS), which handles the models, namely, model generation, selection, training, and testing, i.e., the training data or responds when the API is contacted.

In addition, the HMI module, which is part of the Local Cloud 2, provides support for the visualization of data as well as for system management. The HMI allows us to view historical and live data as they were received from the middleware and inspects the results from the data analysis component (alarms for unusual data, warnings of impending failures, etc.).

This solution builds on the solution proposed in [35] and improves it by leveraging the Arrowhead Framework, to provide to our prototype which is more capable of adapting to changes in the infrastructure, thanks to the orchestrator and system registry services. The prototype also enjoys improved security capabilities, with stronger authentication, and it is capable of interoperating with other technologies, allowing the proposed architecture to evolve easily, and to support other systems and functionalities. The architecture, based on a multiple Arrowhead local cloud, allows to perform analytics-related actions remotely with respect to data collection, and finally, the implementation of the solution was made less challenging and expensive since the Arrowhead Framework is made up of open source components.

5.2. Condition Monitoring of Rolling Element Bearings

The authors present a prototype capable of executing the condition monitoring of rolling-element bearing, which is supported by the architecture defined in Figure 5. Aligned with the discussion at the beginning of Section 5, this condition-monitoring prototype is written in Python language, and it monitors the received time-domain data of the rolling element bearing, not only in the time domain but also in the frequency domain, after performing a fast Fourier transform (FFT) of the time domain signal. Nevertheless, the solution is at the initial phases; In the near future, this prototype will be able to make some analytics such us RUL calculation using VTT O&M analytics libraries [36]. The software's objective is to be integrated into the Arrowhead Framework, i.e., as a service [26], as was mentioned in the previous section. Once it is compliant with the Arrowhead Framework, it is also possible to integrate it with applications that are based on sensors or other IoT devices like, for instance, Arduino [37], NodeMCU [38] and Raspberry Pi [39].

The data monitored in the prototype were simulated with a Python-based simulation program done by VTT. The user of the simulation software can define different bearing characteristics such as Pitch and ball diameters, the number of balls, contact angle, and select between the ball and roller types. In addition, the user can modify operation conditions: e.g., rotations per minute (RPM) and load. Finally, the program allows to simulate none faulty signals or add one the most typical faults of the rolling element bearings such as ball pass frequency outer (BPFO), ball pass frequency inner (BPFI), ball spin frequency (BSF) and fundamental train frequency (FTF) faults. This simulation program can be Arrowhead Framework compliant as a service consumer to validate the system.

A preliminary simulation test was carried out on the baseline behavior of our system, without adding any of the most typical faults of the bearings. We provide the bearing characteristics and the operating conditions available in Table 1 and the sampling characteristics in Table 2. The output signal in the time domain can be seen in Figure 6, whereas Figure 7 shows the spectrum of the signal.

Table 1. Bearing characteristics and the operating conditions of the simulation carried.

RPM	Pitch Diameter	Ball Diameter	Number of Balls	Contact Angle	Type of Bearing
1500	10 mm	5 mm	10 Units	10 degrees	Ball

Table 2. Characteristics of the simulated signal.

Number of Samples	Sampling Frequency	Frequency Resolution
8192 Units	25,600 Hz	3.125 Hz

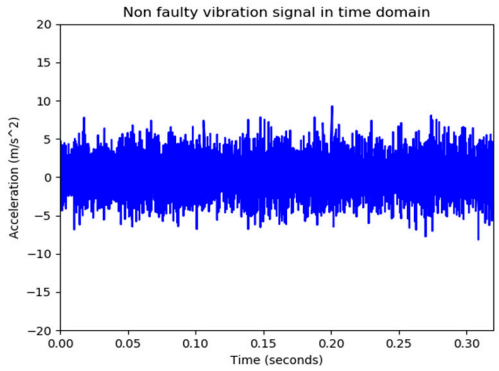

Figure 6. Non-faulty vibration signal in time domain with the characteristics from Table 1.

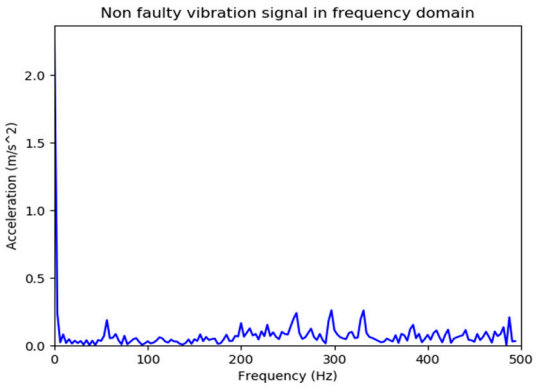

Figure 7. Non-faulty vibration signal in frequency domain with the characteristics of the Table 1.

An additional test was carried out, adding an outer race fault. We used the same simulation characteristics of the first test, and its parameters are available in Tables 1 and 2. The outer race fault simulated signal is the output of the raw vibration signal, plus an impulse caused by a fault on the outer race, which is generated whenever a rolling element passes over the faulty point on outer race. An output signal in the time domain can be seen in Figure 8 and its spectrum in Figure 9.

To understand the results, we calculated the theoretical outer race fault frequency of a bearing simulation using the characteristics described in Table 1. This frequency is 63.4 Hz. Comparing the test carried without adding any fault, it can be observed that Figure 9 includes many peaks of acceleration in the spectrum in multiple frequencies, which includes the theoretical outer race fault frequency at

63.4 Hz and the theoretical inner race fault frequency at 186.6 Hz. Those two values are monitored at 62.5 Hz and 187.5 Hz due to the resolution available in Table 2.

To sum up, the authors have presented a prototype capable of the condition monitoring of a REB with an open source approach, in this case, Python was used to calculate the amplitudes of the fault frequencies, and in the future, these results can be further analyzed using OSA–CBM and Mimosa to save all the results to the database [40].

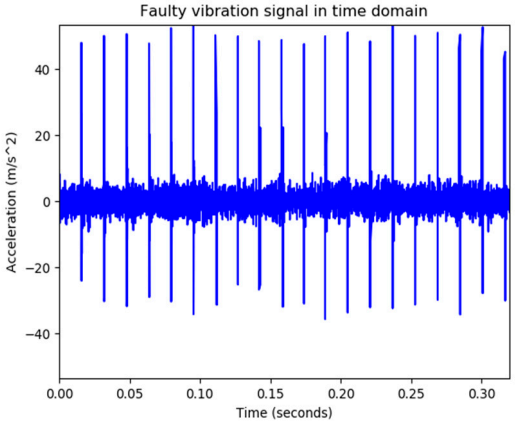

Figure 8. Faulty vibration signal in the time domain with the characteristics in Table 1.

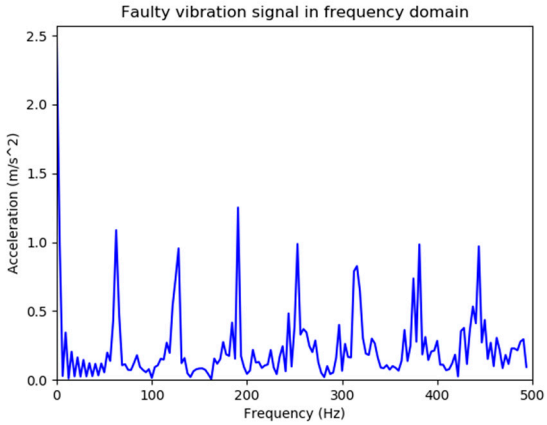

Figure 9. Faulty vibration signal in frequency domain with the characteristics of the Table 1.

6. Conclusions

The use of the Arrowhead Framework provides the possibilities to use all the modules and tools of its ecosystem as well as interoperability between the various systems. It results in options, such as the condition monitoring of geologically distributed machines that belong to a local cloud. Besides, it gives the system the potential use of the service-oriented architecture (SOA) with non-functional requirements, such as the service registry and discovery, as well as security. Thus, it helps in bridging the gap between the maintenance industry and other communities supported by open source technologies. In this sense, the present work is a step further for the democratization of advanced monitoring and maintenance strategies. Further work will be devoted to the design of a core service that contains the model of the machines to be monitored, both for easing the configuration of existing and new local clouds, as well as to allow to communicate and exchange monitoring models.

The Arrowhead Framework provides the data acquisition and collection of multiple sources, such as bearing data or machine status data collected from various I/O ports. This paper, thus, presents the innovative integration between the IoT, in this case, the Arrowhead Framework with the predictive

health monitoring approach that supports the rolling element bearing condition, i.e., rolling element bearing (REB) fault analysis with the support of the Python language. The use of the Arrowhead Framework gives not only the possibilities to add new modules like the one mentioned in the current paper, i.e., REB fault analysis, but also other suitable machine learning algorithms for purposes of condition monitoring and maintenance decision making.

In addition, it considers the best practices and standards in the domain of interest, i.e., OSA–CBM and the MIMOSA CRIS.

The use of Python brings some positive features, such as less development time, which is based on a large number of built-in libraries. Thus, the use of the Python language affects the model reusability, required development efforts and design complexity. Consequently, it is believed that less time and efforts in the development phases reduce related costs. When it comes to less development time, the more favorable development aspects can motivate smaller companies with a limited budget to make investments in systems similar to the one presented in this paper, thus opening the door to employing advanced condition monitoring, prognosis, and decision making aspects for REBs fault analysis system.

Author Contributions: J.C. contributed by conceptualizing the work and article as well as the solution, developing its methodology. He was also involved in writing the original draft as well as reviewing and editing the article. P.S. helped in the development of the methodology, especially the parts related to IoT testing and maintenance, as well as a review of the paper. M.A. contributed to the conceptualization of the solution, the analysis of the benefits of the Arrowhead approach; he worked on the integration with IoT platforms in general and the Arrowhead platform in particular, as well as a review of the article. L.L.F. worked on the Arrowhead platform, as well as a review of the paper. M.L., created the rolling element simulated data as well as performed the analysis and results of it. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The present work has been part of the MANTIS Cyber-Physical System based Proactive Collaborative Maintenance project, which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 662189. The work was further developed in the context of the Productive 4.0 project, which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737459. The ECSEL Joint Undertaking receives support from the European Union's Horizon 2020 research program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Delsing, J.; Varga, P.; Ferreira, L.L.; Albano, M.; Pereira, P.P.; Eliasson, J.; Carlsson, O.; Derhamy, H. 3 The Arrowhead Framework architecture. In *IoT Automation*; Informa UK Limited: London, UK, 2017; pp. 43–88.
2. Turban, E.; Sharda, R.; Denlen, D. *Decision Support and Business Intelligence Systems*, 9th ed.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2011.
3. Thurston, M.G. An open standard for Web-based condition-based maintenance systems. In Proceedings of the 2001 IEEE Autotestcon Proceedings. IEEE Systems Readiness Technology Conference, Valley Forge, PA, USA, 20–23 August 2001.
4. Moubray, J. *Reliability Centred Maintenance*, 2nd ed.; Butterworth-Heinemann Ltd.: Great Britain, UK, 2005; ISBN 0750633581.
5. Jantunen, E.; Hooghoudt, J.-O.; Yang, Y.; McKay, M. Predicting the remaining useful life of rolling element bearings. In Proceedings of the 2018 IEEE International Conference on Industrial Technology (ICIT), Lyon, France, 19–22 February 2018.
6. Campos, J. Development in the application of ICT in condition monitoring and maintenance. *Comput. Ind.* **2009**, *60*, 1–20. [CrossRef]
7. Campos, J. Managing the information systems in the industrial domain. *Cogent Bus. Manag.* **2016**, *3*, 1180967. [CrossRef]
8. Jantunen, E.; Campos, J.; Sharma, P.; Baglee, D. Digitalization of Maintenance. In Proceedings of the 2017 2nd International Conference on System Reliability and Safety (ICSRS), Milan, Italy, 20–22 December 2017; pp. 343–347.

9. El-Thalji, I.; Jantunen, E. A summary of fault modelling and predictive health monitoring of rolling element bearings. *Mech. Syst. Signal Process.* **2015**, *60*, 252–272. [CrossRef]
10. Jantunen, E.; Campos, J.; Sharma, P.; McKay, M. Open Source Analytics Solutions for Maintenance. In Proceedings of the 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), Thessaloniki, Greece, 10–13 April 2018.
11. MIMOSA OSA-CBM. Open System Architecture for Condition-Based Maintenance. Available online: www.mimosa.org/mimosa-osa-cbm/ (accessed on 10 September 2020).
12. Albano, M.; Jantunen, E.; Papa, G.; Zurutuza, U. *The MANTIS Book. Cyber-Physical System Based Proactive Collaborative Maintenance*; River Publisher: Gistrup, Denmark, 2018.
13. Jardine, A.K.; Lin, D.; Banjevic, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mech. Syst. Signal Process.* **2006**, *20*, 1483–1510. [CrossRef]
14. Vlok, P.J.; Wnek, M.; Zygmunt, M. Utilizing statistical residual life estimates of bearings to quantify the influence of preventive maintenance actions. *Mech. Syst. Signal Proc.* **2004**, *18*, 833–847. [CrossRef]
15. Wang, W.Q.; Golnaraghi, M.; Ismail, F. Prognosis of machine health condition using neuro-fuzzy systems. *Mech. Syst. Signal Process.* **2004**, *18*, 813–831. [CrossRef]
16. Dong, Y.-L.; Gu, Y.-J.; Yang, K.; Zhang, W.-K. A combining condition prediction model and its application in power plant. In Proceedings of the Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat No 04EX826) ICMLC-04, Shanghai, China, 26–29 August 2004.
17. Song, H.; Srinivasan, R.; Sookoor, T.; Jeschke, S. *Smart Cities: Foundations, Principles, and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2017.
18. Jeschke, S.; Brecher, C.; Meisen, T.; Özdemir, D.-I.D.; Eschert, T. Industrial Internet of Things and Cyber Manufacturing Systems. In *Springer Series in Wireless Technology*; Springer Science and Business Media LLC: Berlin, Germany, 2016; pp. 3–19.
19. Song, H.; Rawat, D.B.; Jeschke, S.; Brecher, C. *Cyber-Physical Systems: Foundations, Principles and Applications*; Morgan Kaufmann: Burlington, MA, USA, 2016.
20. Gartner Inc. Top Trends in the Gartner Hype Cycle for Emerging Technologies. 2017. Available online: https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/ (accessed on 10 September 2020).
21. Coetzee, L.; Eksteen, J. The Internet of Things—Promise for the Future? An introduction. In Proceedings of the IST-Africa Conference, Gaborone, Botswana, 11–13 May 2011; pp. 1–9, ISBN 978-1-905824-24-3.
22. Gyrard, A.; Datta, S.K.; Bonnet, C.; Boudaoud, K. Cross-Domain Internet of Things Application Development: M3 Framework and Evaluation. In Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, Italy, 24–26 August 2015.
23. Vögler, M.; Schleicher, J.M.; Inzinger, C.; Dustdar, S. A Scalable Framework for Provisioning Large-Scale IoT Deployments. *ACM Trans. Internet Technol.* **2016**, *16*, 1–20. [CrossRef]
24. Kim, H.; Ahmad, A.; Hwang, J.; Baqa, H.; Le Gall, F.; Ortega, M.A.R.; Song, J. IoT-TaaS: Towards a Prospective IoT Testing Framework. *IEEE Access* **2018**, *6*, 15480–15493. [CrossRef]
25. Pontes, P.M.; Lima, B.; Faria, J.P. Izinto: A pattern-based IoT testing framework. In Proceedings of the Companion Proceedings for the ISSTA/ECOOP 2018 Workshops, Amsterdam, The Netherlands, 16–21 July 2018.
26. Jararweh, Y.; Al-Ayyoub, M.; Darabseh, A.; Benkhelifa, E.; Vouk, M.; Rindos, A. SDIoT: A software defined based internet of things framework. *J. Ambient. Intell. Humaniz. Comput.* **2015**, *6*, 453–461. [CrossRef]
27. Paniagua, C.; Delsing, J. Industrial Frameworks for Internet of Things: A Survey. *IEEE Syst. J.* **2020**, 1–11. [CrossRef]
28. Derhamy, H.; Eliasson, J.; Delsing, J.; Priller, P. A survey of commercial frameworks for the Internet of Things. In Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Parc Hotel Alvisse, Luxembourg, 8–11 September 2015; pp. 1–8.
29. Blomstedt, F.; Ferreira, L.L.; Klisics, M.; Chrysoulas, C.; De Soria, I.M.; Morin, B.; Zabasta, A.; Eliasson, J.; Johansson, M.; Varga, P. The Arrowhead Approach for SOA Application Development and Documentation. In Proceedings of the IECON 2014—40th Annual Conference of the IEEE Industrial Electronics Society, Dallas, TX, USA, 29 October–1 November 2014.
30. Pedersen, T.; Albano, M.; Nielsen, B. Reengineering the lifecycle of Arrowhead applications: From skeletons to the client library. In Proceedings of the IECON 2019—45th Annual Conference of the IEEE Industrial Electronics Society, Lisbon, Portugal, 14–17 October 2019.

31. Albano, M.; Nielsen, B. Interoperability by construction: Code generation for Arrowhead Clients. In Proceedings of the 3rd IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2020), Tampere, Finland, 10–12 June 2020.
32. Basissystem Industrie 4.0. Available online: https://www.sysgo.com/company/about-sysgo/rd-projects/basys-40 (accessed on 10 September 2020).
33. FIWARE (FIWARE Foundation, Core Platform for the Future Internet). Available online: https://www.fiware.org/about-us/(accessed on 10 September 2020).
34. Vyatkin, V. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*; Instrument Society of America: Champaign, IL, USA, 2007.
35. Ferreira, L.L.; Albano, M.; Silva, J.; Martinho, D.; Marreiros, G.; Di Orio, G.; Maló, P.; Ferreira, H. A pilot for proactive maintenance in industry 4.0. In Proceedings of the 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), Trondheim, Norway, May 31–June 2 2017.
36. Larrañaga, M.; Salokangas, R.; Kaarmila, P.; Saarela, O. Low-Cost Solutions for Maintenance with a Raspberry Pi. In Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference, Venice, Italy, 1–6 November 2020; Research Publishing Services: Singapore, 2020.
37. Koenka, I.J.; Sáiz, J.; Hauser, P.C. Instrumentino: An open-source modular Python framework for controlling Arduino based experimental instruments. *Comput. Phys. Commun.* **2014**, *185*, 2724–2729. [CrossRef]
38. Hastbacka, D.; Jaatinen, A.; Hoikka, H.; Halme, J.; Larranaga, M.; More, R.; Mesia, H.; Bjorkbom, M.; Barna, L.; Pettinen, H.; et al. Dynamic and Flexible Data Acquisition and Data Analytics System Software Architecture. In Proceedings of the 2019 IEEE SENSORS, Montreal, Canada, 27–30 October 2019.
39. Halme, J.; Jantunen, E.; Hastbacka, D.; Hegedus, C.; Varga, P.; Bjorkbom, M.; Mesia, H.; More, R.; Jaatinen, A.; Barna, L.; et al. Monitoring of Production Processes and the Condition of the Production Equipment through the Internet. In Proceedings of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 23–26 April 2019.
40. Salokangas, R.; Jantunen, E.; Larrañaga, M.; Kaarmila, P. Mimosa Strong Medicine for Maintenance. In *Advances in Wireless Communications and Applications*; Springer Science and Business Media LLC: Berlin, Germany, 2020; pp. 35–47.